JOURNAL OF COMPLEXITY 7, 408–424 (1991)

On the Evaluation of the Eigenvalues of a Banded Toeplitz Block Matrix

DARIO BINI

Dipartimento di Matematica, Università di Pisa

AND

VICTOR PAN*

Department of Mathematics and Computer Science, Lehman College, City University of New York, and Department of Computer Science, State University of New York at Albany

Received July 26, 1988

Let $A = (a_{ij})$ be an $n \times n$ Toeplitz matrix with bandwidth $k + 1$, $k = r + s$, that is, $a_{ij} = a_{j-i}$, $i, j = 1, \ldots, n$, $a_i = 0$ if $i > s$ and if $i < -r$. We compute $p(\lambda) = \det(A - \lambda I)$, as well as $p(\lambda)/p'(\lambda)$, where $p'(\lambda)$ is the first derivative of $p(\lambda)$, by using $O(k \log k \log n)$ arithmetic operations. Moreover, if a_i are $m \times m$ matrices, so that A is a banded Toeplitz block matrix, then we compute $p(\lambda)$, as well as $p(\lambda)/p'(\lambda)$, by using $O(m^3 k(\log^2 k + \log n) + m^2 k \log k \log n)$ arithmetic operations. The algorithms can be extended to the computation of $\det(A - \lambda B)$ and of its first derivative, where both A and B are banded Toeplitz matrices. The algorithms may be used as a basis for iterative solution of the eigenvalue problem for the matrix A and of the generalized eigenvalue problem for A and B.

1. INTRODUCTION

The problem of computing the eigenvalues of an $n \times n$ banded (block) Toeplitz matrix $A = (a_{j-i})$ with bandwidth $k + 1$, where $k = r + s$, $a_i = 0$ if $i > s$ and if $i < -r$, $a_s a_{-r} \neq 0$, has been recently considered by several authors. This problem can be numerically solved by applying any root-finding method to the characteristic equation $p(\lambda) = \det(A - \lambda I) = 0$, so that efficient algorithms for the evaluation of $p(\lambda)$ or for both $p(\lambda)$ and

* Supported by NSF Grant CCR 8805782.

0885-064X/91 $3.00

$p(\lambda)/p'(\lambda)$ are particularly useful in this case (here $p'(\lambda)$ denotes the first derivative of the polynomial $p(\lambda)$ with respect to λ). If the ratio $p(\lambda)/p'(\lambda)$ is easily computable, then Newton's iteration method can be applied as root-finder, and we may take advantage of its quadratic convergence (see the Remark at the end of this section).

In this context it is very important to devise efficient algorithms for the evaluation of the characteristic polynomial

$$p(\lambda) = p(a_{-r}, \ldots, a_s, \lambda) = \det(A - \lambda I) \tag{1}$$

and of the ratio $p(\lambda)/p'(\lambda)$, where

$$p'(\lambda) = q(a_{-r}, \ldots, a_s, \lambda) = \partial p(a_{-r}, \ldots, a_s, \lambda)/\partial\lambda \tag{2}$$

given $\{a_{-r}, \ldots, a_s, \lambda\} \cup \mathbf{C}$, where $\mathbf{C}$ is a field of constants (to be certain, we let it be the complex field), in the set of rational functions in the indeterminates $a_{-r}, \ldots, a_s, \lambda$ (Borodin and Munro, 1975). We estimate the cost of these two algorithms under the sequential and parallel Random Access Machine (RAM, PRAM, respectively) models of arithmetic computations (Aho *et al.*, 1976; Eppstein and Galil, 1988; Borodin *et al.*, 1982; Karp and Ramachandran, 1990) with the uniform cost measure given by the number of arithmetic operations involved (in the sequential case) and by the numbers of parallel arithmetic steps and processors needed in the parallel case. In the PRAM model we assume that in each step at most p arithmetic operations can be simultaneously performed, where p is the number of processors, and we will say that pt is the total work or the sequential time of a parallel t-step algorithm performed using p processors. We say that an algorithm computes a function $p(a_{-r}, \ldots, a_s, \lambda)$, depending on k and n, with cost $O(f(n, k))$ if there exists a constant c such that for any n and k and for any input $a_{-r}, \ldots, a_s, \lambda$, the algorithm computes $p(a_{-r}, \ldots, a_s, \lambda)$ with cost at most $cf(n, k)$.

Even though our main and final objective is approximating the eigenvalues, there are some advantages of staying in the domain of rational computations at the intermediate stage of computing the characteristic polynomial (whose coefficients lie in the ring of the matrix entries and, in particular, are integers if the entries are integers). Specifically, our rational algorithms can be immediately extended to the highly important case where the input entries are blocks, that is, where we deal with block Toeplitz input matrices.

In Bini and Capovani (1983a; 1983b) some methods are presented for computing (1) and (2) where A is a real symmetric or a complex Hermitian matrix. Their sequential arithmetic cost (their total work) is high; they are in NC (see Eppstein and Galil, 1988; Karp and Ramachandran, 1990 on NC), and their parallel arithmetic time is only $O(\log n + \log^2 k)$ under the

PRAM model. These algorithms are based on the technique of matrix embedding, and a generalization of this approach is presented in Bini (1985) (compare also Bini and Capovani, 1987). The high sequential arithmetic cost of the algorithms of Bini and Capovani (1983a; 1983b) is substantially reduced in Trench (1991b; 1985) for the price of changing the model of computation. Instead of computing the polynomial of (1), the algorithm of Trench (1985) (see also Trench, 1991b) approximates this polynomial with an absolute error at most ε, that is, computes a value $\tilde{p}(\lambda)$ such that $|\tilde{p}(\lambda) - p(\lambda)| \leq \varepsilon$.

The sequential arithmetic cost of this algorithm is $O(k \log n + k^3 + C(k, \eta))$, where $C(k, \eta)$ is the cost of approximating to all the zeros of an auxiliary polynomial of degree k with absolute errors at most η (see the Table I in Section 2 where the record complexity estimates from Pan, 1987, are given for the root-finding problem). The dependence of η on the output error bound ε, k, and n has not yet been analyzed. Such an analysis would require to study the numerical conditioning of the output as a function of the zeros of the auxiliary polynomial.

In such cases the arithmetic operations count does not fully represent the complexity of the computational problem and in fact can be misleading in some cases. For instance, by operating with sufficiently long integers, we may multiply or divide two polynomials with integer coefficients or multiply two matrices with integer entries by performing only a single multiplication (Bini and Pan, 1986; Pan, 1984, Section 40). The algorithm of Trench (1991b) should not be discarded on this ground, however. For instance, the numerical stability problems are not likely to arise in the important cases where k is small, and then the arithmetic count above is meaningful.

Due to the stage of computing polynomial zeros, the algorithm of Trench (1985), however, only outputs an approximate (rather than exact) value of $p(\lambda)$, does not approximate to $p(\lambda)/p'(\lambda)$, and does not apply in the case where A is a block matrix.

Technically, the algorithm of Trench (1985) relies on the reduction of the problem to a linear difference equation, which is solved by its reduction to the associated characteristic equation.

In Bini and Pan (1988), it is shown how to keep the computation in the exact model, avoiding approximation and only slightly increasing the overall cost, that is, to $O(k \log k \log n + k^3)$. The idea is to solve the difference equation associated with the original problem by powering a Frobenius matrix and by using fast polynomial arithmetic. Unlike Trench (1991b; 1985), this approach can be easily extended to the computation of $p(\lambda)/p'(\lambda)$ and to the case where the matrix A is a block matrix (the importance of an extension to the case of block matrices has been pointed out also in Tismenetsky, 1987, but no algorithms for this problem have been presented in Tismenetsky, 1987).

In this paper, we devise a new method for the evaluation of both $p(\lambda)$ and $p(\lambda)/p'(\lambda)$, which substantially improves the known complexity results. Our improvement stems from a simple but general and rather surprising reduction of the original problem to the computations for an auxiliary $s \times s$ Toeplitz matrix; to yield such a reduction and to make use of it, we had to incorporate and modify several techniques of Bini and Capovani (1983a; 1983b), Bini and Pan (1988), and Trench (1985).

We implement our method in the form of three distinct approaches. The simplest approach gives Algorithm 1 below, which we present mostly for demonstration and also because its parallel arithmetic time cost is very small under the PRAM model: $O(\log n + \log^2 k)$ steps for computing both $p(\lambda)$ and $p(\lambda)/p'(\lambda)$, although its sequential arithmetic cost is too high (being of an order higher than n).

In the second implementation, we adopt the model of approximate computation where the zeros of the kth degree polynomial associated to the linear difference equation must be approximated within a fixed absolute error bound η. In this case, we obtain Algorithm 2, which we implement using $O(k \log n + k^2 + C(k, \eta))$ arithmetic operations. (We present this estimate to make a comparison with Trench (1991b), even though we understand the deficiency of the model of that work.) As in Trench (1985), the algorithm cannot be extended to the case of block matrices and does not allow the direct computation of $p(\lambda)/p'(\lambda)$.

The third implementation, obtained by means of the technique of repeated squaring of a Frobenius matrix, yields Algorithm 3, which involves $O(k \log k \log n)$ arithmetic operations and can be implemented under the PRAM model for the cost of $O(\log k \log n)$ parallel steps with k^2 processors. This algorithm can be extended to the case of block matrices for the cost of $O(m^3 k(\log n + \log^2 k) + m^2 k \log k \log n)$ arithmetic operations; here m is the dimension of the blocks.

The above bounds on the computational complexity have been obtained relying on the known asymptotically fast auxiliary algorithms for the computations with polynomials and with general and special matrices over the field of complex numbers.

We present our results in the following order: in Section 2 we recall the cost estimates of the auxiliary algorithms for the computations with polynomials and structured matrices. In Section 3, we present a matrix identity that allows us to reduce the computations for the banded Toeplitz matrix A to the computations for an $s \times s$ triangular Toeplitz matrix. In Section 4, we present the three algorithms (1, 2, and 3) for computing the values of the characteristic polynomial $p(\lambda)$ of a banded Toeplitz matrix together with their cost estimates. This also shows the basic techniques involved. In Section 5 we indicate how to extend the algorithms to computing both $p(\lambda)$ and $p(\lambda)/p'(\lambda)$ in the case where A is a banded Toeplitz

matrix or a banded Toeplitz block matrix. We leave Section 6 for a discussion and some concluding remarks.

Remark. The question of choosing Newton's or, say, the secant method, for function root-finding is quite delicate. It is clear at least that it would be very unwise to discard Newton's method for *polynomial* root-finding, in view of the results on its quadratic and simultaneous global convergence, right *from the start* (Smale, 1986; Renegar, 1987). These results have recently been shown to be highly powerful for the matrix eigenvalue computation too (Bini and Pan, 1990, 1991).

The authors thank the referees for helpful comments.

2. Preliminaries

The Table I summarizes some upper estimates for the arithmetic complexity of certain computations with polynomials and structured matrices. In this table, $q(x)$ and $r(x)$ denote the quotient and the remainder of the division of two input polynomials, $u(x)$ and $v(x)$, of degrees at most n; A, T, T_k, and Ω denote $n \times n$ matrices, where A is a Toeplitz matrix, T is a triangular Toeplitz matrix, and T_k is a triangular Toeplitz matrix with the bandwidth at most $k + 1$; the evaluation of all the zeros of $u(x)$ (having coefficients less than 2^m for a fixed integer m) means their approximating within $\eta = 2^{-b}$; this specifically means that we must perform, with the (lowest possible) precision of $O(n(b + m))$ bits, the computation that output n numbers whose distances from the corresponding zeros (tolerance to the absolute errors of the output) are bounded by $\eta = 2^{-b}$ where $b < m$ is a fixed integer; and Ω is the transpose of an $n \times n$ Vandermonde matrix.

We will call a matrix *strongly nonsingular* if all of its leading principal

TABLE I

Evaluation of	Arithmetic operations	Parallel steps	Processors	Ref.
$q(x)$ and $r(x)$	$O(n \log n)$	$O(\log n)$	n^2	Bini (1984); Bini and Pan (1986)
		$O(\log n \log\log n)$	$n/\log\log n$	Reif and Tate (1989)
$w(x) = u(x)v(x)$	$O(n \log n)$	$O(\log n)$	n	Borodin and Munro (1975)
All zeros of $u(x)$	$O(n^2 \log n \log b)$	$O(n \log n \log(bn))$	$n \log b/\log(bn)$	Pan (1987)
T^{-1}	$O(n \log n)$	$O(\log n)$	n^2	Bini (1984); Bini and Pan (1986)
		$O(\log n \log\log n)$	$n/\log\log n$	Reif and Tate (1989)
T_k^{-1}	$O(n \log n)$	$O(\log n)$	nk	Bini (1984)
$\Omega_n \mathbf{u}$, $\Omega_n^{-1}\mathbf{u}$	$O(n \log^2 n)$	$O(\log^2 n)$	$n^2/\log n$	Canny *et al.* (1989)

submatrices are nonsingular; for a strongly nonsingular matrix there exists its unique LU factorization where L is a unit lower triangular matrix. The set of the matrices that are not strongly nonsingular is a zero-measure subset and, moreover, an algebraic manifold of a lower dimension in the space of all the $n \times n$ matrices.

We also need the following auxiliary result:

PROPOSITION 1. *Given an $n \times n$ Toeplitz matrix A, the value* det A *can be computed with $O(n \log^2 n)$ arithmetic operations if A is strongly nonsingular; otherwise, $O(n^2 \log n)$ arithmetic operations are sufficient to compute* det A. *In the parallel model of computations $O(\log^2 n)$ arithmetic steps with $O(n^2/\log n)$ processors are sufficient to compute* det A *for any $n \times n$ Toeplitz matrix.*

Proof. If A is strongly nonsingular then there exists the LU factorization of A. The diagonal entries of U can be computed by means of the generalized Schur algorithm having the arithmetic cost $O(n \log^2 n)$ Ammar and Gragg (1987). For a general Toeplitz matrix A it is sufficient to apply the algorithm presented in Pan (1991) that computes det A in $O(\log^2 n)$ steps with $O(n^2/\log n)$ processors. ∎

We may relax the strong nonsingularity assumption if we only seek $|det\ A|$. Indeed, we may compute the diagonal entries of the U factor in the LU factorization of $A^T A$ (see Ammar and Gragg, 1987) and then compute $\det(A^T A) = |det\ A|^2$ in $O(n \log^2 n)$ arithmetic operations, for $A^T A$ is a strongly nonsingular Toeplitz-like matrix. Also, $|\det A|$ is given by the QR factorization of A, which can be computed in $O(n^2)$ arithmetic operations.

The algorithms for polynomial division and polynomial multiplication that give the cost bounds of Table I, as well as the algorithms for inverting a triangular Toeplitz matrix and for computing the determinant of a general Toeplitz matrix, can be easily extended to the case of matrix polynomials (i.e., polynomials having matrix coefficients) and to the case of block Toeplitz matrices (compare Bini and Pan, 1988). If m is the size of the blocks or of the matrix coefficients of the polynomials, then we have the following sequential arithmetic cost bounds (see Table II).

TABLE II

Evaluation of	Arithmetic operations
$q(x)$ and $r(x)$	$O(m^3 n + m^2 n \log n)$
$w(x) = u(x)v(x)$	$O(m^3 n + m^2 n \log n)$
T^{-1}	$O(m^3 n + m^2 n \log n)$
det A	$O(m^3 n^2 \log^2 n)$ $(O(m^3 n \log n))$

In this table, u, v, and w are matrix polynomials, q and r are the right quotient and the right remainder of the division of u and v, i.e., $u(x) = v(x)q(x) + r(x)$, degree$(r(x)) <$ degree$(v(x))$, T is a block triangular block Toeplitz matrix, A is a block Toeplitz matrix, and the cost bound between brackets in the fourth row applies if the block Toeplitz matrix has a block LU factorization.

3. The Basic Reduction of the Problem

Hereafter, the subscripts of the block matrices will denote their size; for square blocks we will write s, n rather than $s \times s$, $n \times n$. We will rely on the following simple fact:

Fact 1. Let a pair of $(n + s) \times (n + s)$ nonsingular matrices T and T^{-1} be represented as 2×2 block matrices as follows:

$$T = \begin{pmatrix} M_{s\times n} & N_s \\ P_n & Q_{n\times s} \end{pmatrix}, \qquad T^{-1} = \begin{pmatrix} U_{n\times s} & W_n \\ V_s & Z_{s\times n} \end{pmatrix}.$$

Then

$$\begin{pmatrix} 0_{s\times n} & I_s \\ P_n & Q_{n\times s} \end{pmatrix} = \begin{pmatrix} V_s & Z_{s\times n} \\ 0_{n\times s} & I_n \end{pmatrix} T,$$

whence

$$(-1)^{s(n+s+1)} \det P_n = \det V_s \det T,$$
$$P_n^{-1} = W_n - U_{n\times n} V_s^{-1} Z_{s\times n},$$

where 0 and I denote the null and the identity matrices, whose size is shown by the subscripts.

Fact 1 can be used in order to reduce the evaluation of the determinant (and of the inverse) of an $n \times n$ banded matrix $A = (a_{ij})$ such that $a_{ij} = 0$ if $j - i \geq s$ to the evaluation of the determinant (and of the inverse) of the $s \times s$ matrix V_s. The reduction is by means of embedding the matrix A as a block P_n into an $(s + n) \times (s + n)$ triangular matrix T.

In the case where A is a banded Toeplitz matrix, such a reduction was effectively exploited in Bini (1984) and Bini and Capovani (1987). We will follow this line. In particular, we will embed a given $(r + s + 1)$-banded Toeplitz matrix $A = (a_{j-i})$ of size $n \times n$, $a_i = 0$ if $i > s$ or $i < -r$, $a_s a_{-r} \neq 0$, into a lower triangular Toeplitz matrix T of size $(n + s) \times (n + s)$ whose

first column is the $(n + s)$-vector $(a_s, a_{s-1}, \ldots, a_{-r}, 0, \ldots, 0)^T$,

$$T = \begin{pmatrix} T_s & 0_{s\times(n-s)} & 0_s \\ & A & 0_{(n-s)\times s} \\ & & T_s \end{pmatrix},$$

where T_s is a lower triangular Toeplitz matrix. Here and throughout the paper, we assume $s \leq r$.

For instance, for $s = r = 2$ and $n = 4$ we have the embedding

$$\begin{pmatrix} a_2 & & & & & \\ a_1 & a_2 & & & & \\ a_0 & a_1 & a_2 & & & \\ a_{-1} & a_0 & a_1 & a_2 & & \\ a_{-2} & a_{-1} & a_0 & a_1 & a_2 & \\ 0 & a_{-2} & a_{-1} & a_0 & a_1 & a_2 \end{pmatrix}.$$

Fact 1 implies that

$$\det A = (-1)^{(n+s+1)s} \det T \det V_s = (-1)^{(n+s+1)s} a_s^{n+s} \det V_s. \tag{3}$$

Moreover, T^{-1} is a lower triangular Toeplitz matrix, since T is a nonsingular lower triangular Toeplitz matrix; in fact, the set of lower triangular Toeplitz matrices is a matrix algebra. Thus, the problem of computing $\det A$ is reduced to that of computing the determinant of the $s \times s$ Toeplitz matrix V_s. The same approach applies to computing $p(\lambda) = \det(A - \lambda I)$ at any point λ since $A - \lambda I$ remains a $(k + 1)$-diagonal Toeplitz matrix.

4. The Algorithms

Here is the first (and the most straightforward but not the most effective) implementation of this approach:

Algorithm 1 (Parallel Implementation). *Stage* 1. Compute T^{-1} by solving the triangular Toeplitz system $T\mathbf{x} = \mathbf{e}_1$, where $\mathbf{e}_1^T = (1, 0, \ldots, 0)$, for the cost of $O(\log(n + s))$ steps using $n(k + 1)$ processors. (T^{-1} is a triangular Toeplitz matrix, defined by $\mathbf{x} = T^{-1}\mathbf{e}_1$, its first column.)

Stage 2. Compute $\det V_s$ using $O(\log^2 s)$ steps and $s^2/\log s$ processors. (V_s is an $s \times s$ Toeplitz matrix.)

Stage 3. Compute $\det A = a_s^{n+s}d$ by means of repeated squaring by using $1 + [\log_2(n + s)]$ steps on a single processor.

The overall number of parallel steps required by the latter algorithm is quite small, that is, $O(\log n + \log^2 s)$, but the number of processors, and therefore, the total arithmetic work, is very large.

Trying to improve Algorithm 1, we observe that we do not need to compute all the entries of T^{-1} in order to compute V_s, but it is sufficient to compute the last $2s - 1$ components of the vector $\mathbf{x}$ such that

$$T\mathbf{x} = \mathbf{e}_1, \qquad \mathbf{e}_1 = (1, 0, \ldots, 0)^T,$$

that is, the last $2s - 1$ components of the first column of T^{-1}, which are also the $2s - 1$ components of the first row and the first column of the $s \times s$ Toeplitz matrix V_s. Due to the Toeplitz structure of T, this system can be viewed as the following linear difference equation:

$$\sum_{j=-r}^{s} a_j x_{j+i-s} = 0, \qquad i = 2, \ldots, n + s, \tag{4}$$

where the auxiliary variables satisfy the initial conditions

$$x_1 = a_s^{-1}, \qquad x_0 = x_{-1} = \ldots = x_{-k+2} = 0. \tag{5}$$

We exploit this representation in the two algorithms that follow.

Expressing the solution of the homogeneous difference equation in terms of the zeros $\alpha_1, \alpha_2, \ldots, \alpha_k$ of the associated polynomial $\varphi(z) = \sum_{i=0}^{k} z^i a_{i-r}$ and in terms of the new unknowns $\xi_1, \ldots, \xi_k$ (see Stoer and Bulirsch, 1980) we have

$$\mathbf{x} = \Omega \begin{pmatrix} \xi_1 \\ \xi_2 \\ \cdot \\ \cdot \\ \xi_k \end{pmatrix}, \qquad \Omega = (\alpha_j^i), i = 0, 1, \ldots, n + s + k - 2, j = 1, 2, \ldots, k,$$

so that $\Omega = \Omega_{k\times(n+s+k-1)}$ is the $k \times (n + s + k - 1)$ transposed Vandermonde matrix (here we suppose for simplicity that $\alpha_i \neq \alpha_j$ **if** $i \neq j$; otherwise, see Stoer and Bulirsch, 1980; Trench, 1985).

Imposing the initial conditions (5) implies that

$$\Omega_k \begin{pmatrix} \xi_1 \\ \xi_2 \\ \cdot \\ \cdot \\ \cdot \\ \xi_k \end{pmatrix} = \begin{pmatrix} x_{-k+2} \\ x_{-k+3} \\ \cdot \\ \cdot \\ \cdot \\ x_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ a_s^{-1} \end{pmatrix} \tag{6}$$

where Ω_k is the $k \times k$ transposed Vandermonde matrix, $\Omega_k = (\alpha_j^i)_{i=0,\ldots,k-1,\, j=1,\ldots,k}$. The last $k = s + r \geq 2s$ components of the vector $\mathbf{x}$ are given by

$$\begin{pmatrix} x_{n-r+1} \\ \cdot \\ \cdot \\ \cdot \\ x_{n+s} \end{pmatrix} = \Omega_k \operatorname{Diag}(\alpha_1^\sigma, \alpha_2^\sigma, \ldots, \alpha_k^\sigma) \begin{pmatrix} \xi_1 \\ \xi_2 \\ \cdot \\ \cdot \\ \cdot \\ \xi_k \end{pmatrix}, \qquad \sigma = n + s - 1. \tag{7}$$

Now, we are ready to present the following algorithm, which approximates $\det A$ with any desired precision:

ALGORITHM 2 (Approximation Algorithm). *Stage* 1. Approximate, up to within absolute error bound 2^{-b}, the zeros $\alpha_1, \alpha_2, \ldots, \alpha_k$ of the polynomial $\varphi(z)$.

Stage 2. Solve the system (6) (with the transposed Vandermonde coefficient matrix Ω_k).

Stage 3. Compute $x_{n-s+2}, \ldots, x_{n+s}$ in (7) obtaining V_s.

Stage 4. Compute $\det A = (-1)^s a_s^{n+s} \det V_s$ (V_s is a Toeplitz matrix).

We observe that $\det A$, as a function of $\alpha_1, \alpha_2, \ldots, \alpha_k$, may be numerically ill-conditioned; that is, a small approximation error in the zeros $\alpha_1, \alpha_2, \ldots, \alpha_k$, may imply a large error in the result. This means that in the complete complexity analysis of this algorithm we should estimate how small 2^{-b} must be (where b is a function $b = b(\varepsilon, n, k)$) in order to have absolute error at most ε in the result. Here, we simply present the arithmetic cost of the algorithm as a function in n, k, and b.

Algorithm 2 can be performed in $O(k \log n + k^2 \log k \log(bk))$ arithmetic operations, which may be reduced to $O(k \log n + k^2 \log k \log b)$ if the matrix V_s is strongly nonsingular. Specifically, by using Proposition 1 and Table I, we obtain that we need $O(k^2 \log k \log b)$ operations at stage 1; $O(k \log^2 k)$ operations to solve the transposed Vandermonde system at stage 2; $O(k \log n)$ operations in order to compute α_i^σ for all i (by means of repeated squaring) at stage 3; $O(k \log^2 k)$ in order to compute the product of the transposed Vandermonde matrix Ω_k by a vector, also at stage 3;

$O(\log n)$ for computing a_s^{n+s} (by means of repeated squaring) at stage 4; $O(s^2 \log^2 s)$ for computing $\det V_s$, also at stage 4, which can be reduced to $O(s \log^2 s)$ if V_s is strongly nonsingular.

Analogously, it is possible to check that, under the parallel arithmetic model of computation, the algorithm has the following arithmetic parallel cost: $O(\log n + k \log k \log(bk))$ steps with $k^2/\log k$ processors.

Finally we present our third algorithm, exploiting the expression of Bini and Pan (1988) for the solution of the linear difference equation in terms of the following $k \times k$ Frobenius matrix:

$$F = \begin{pmatrix} 0 & 1 & & & \\ & 0 & \ddots & & \\ & & \ddots & \ddots & \\ & & & 0 & 1 \\ g_0 & g_1 & \cdots & g_{k-2} & g_{k-1} \end{pmatrix},$$

$$g_i = -a_{i-r}/a_s, \quad i = 0, \ldots, k-1.$$

It is easy to check that the solution $\{x_i\}$ of (4) satisfying (5) can be rewritten as

$$\begin{pmatrix} x_{i-k+2} \\ \vdots \\ x_{i+1} \end{pmatrix} = F^i a_s^{-1} \mathbf{e}_k.$$

Here and hereafter, $\mathbf{e}_i$ denotes the ith column of the $k \times k$ identity matrix. Therefore, the last $2s-1$ components of the vector $\mathbf{x}$ are equal to the last $2s-1$ components of the vector

$$\mathbf{u} = F^{n+s-1} a_s^{-1} \mathbf{e}_k. \tag{8}$$

The computation of the vector $\mathbf{u}$ in (8) can be carried out by means of repeated squaring (as in Bini and Pan, 1988) since $\varphi(F) = 0$ for the polynomial $\varphi(z) = \sum_{i=0}^{k} z^i a_{i-r}$. Therefore, $F^{n+s-1} = \psi(F)$ if $\psi(z)$ is the polynomial of degree $k-1$ such that $\psi(z) = z^{n+s-1} \bmod \varphi(z)$. Thus, we may proceed in the following way: first compute the coefficients of the polynomial

$$\psi(z) = \sum_{i=0}^{k-1} \theta_i z^i = z^{n+s-1} \bmod \varphi(z)$$

by means of repeated squaring modulo $\psi(z)$, then compute the vector $\mathbf{u} = a_s^{-1}\psi(F)\mathbf{e}_k$. Toward this purpose, observe that $F^i\mathbf{e}_k = \tilde{T}^{-1}\mathbf{e}_{k-i}$, $i = 0, \ldots, k-1$, where $\tilde{T}$ is the lower triangular $k \times k$ Toeplitz matrix whose first column is $(1, -g_{k-1}, \ldots, -g_1)^T$. Consequently,

$$\mathbf{u} = a_s^{-1}\sum_{i=0}^{k-1}\theta_i F^i\mathbf{e}_k = a_s^{-1}\sum_{i=0}^{k-1}\tilde{T}^{-1}\theta_i\mathbf{e}_{k-i} = a_s^{-1}\tilde{T}^{-1}\boldsymbol{\theta},$$

$$\boldsymbol{\theta} = (\theta_{k-1}, \ldots, \theta_0)^T. \tag{9}$$

Now, we are ready to present our third algorithm, where, for the sake of simplicity, we assume that $n + s - 1 = 2^h$:

ALGORITHM 3 (Exact Solution Algorithm). *Stage* 1. Compute the coefficients θ_j of the polynomial

$$\psi(z) = \sum_{j=0}^{k-1}\theta_j z^j = z^{n+s-1} \bmod \varphi(z)$$

by performing the following steps:

$$\psi_0(z) = z,$$
$$\psi_{i+1}(z) = \psi_i^2(z) \bmod \varphi(z), \qquad i = 0, 1, \ldots, h-1,$$
$$\psi(z) = \psi_h(z).$$

Stage 2. Compute the vector $\mathbf{u}$ of (8) by solving the $k \times k$ triangular Toeplitz system

$$\tilde{T}\mathbf{u} = a_s^{-1}\boldsymbol{\theta},$$

where $\tilde{T}$ is the lower triangular $k \times k$ Toeplitz matrix whose first column is $(1, -a_{s-1}/a_s, \ldots, -a_0/a_s)^T$ and $\boldsymbol{\theta} = (\theta_{k-1}, \ldots, \theta_0)^T$ (compare (9)). This way we obtain the last $2s - 1$ components of the vector $\mathbf{x}$ satisfying (4) and (5), which determines the Toeplitz matrix V_s.

Stage 3. Compute $(-1)^s a_s^{n+s}\det V_s$.

By using the results of Section 2 we may easily check that the arithmetic cost of Algorithm 3 is given by: $O(k \log k \log n)$ at stage 1, $O(k \log k)$ at stage 2, $O(\log n + s^2 \log s)$ at stage 3 (reduced to $O(\log n + s \log^2 s)$ if the matrix V_s is strongly nonsingular), and $O(k(\log k)(\log n + k))$ in the entire algorithm (reduced to $O(k \log k \log n)$ if the matrix V_s is strongly nonsingular).

Under the parallel arithmetic model of computation Algorithm 3 requires $O(\log k \log n)$ parallel steps and $k^2/\log k$ processors.

Since the matrix $A - \lambda I$ is still a $(k+1)$-diagonal Toeplitz matrix, algorithms 1, 2, and 3 can be applied to compute the value that the polynomial $p(\lambda) = \det(A - \lambda I)$ takes on at a point λ. In this case a_0 must be replaced by $a_0 - \lambda$. Moreover, these algorithms can also be used to compute $\det(A - \lambda B)$ in the algorithms for the generalized eigenvalue problem, $A\mathbf{x} = \lambda B\mathbf{x}$, where A and B are band Toeplitz matrices (in this case, r and s are the greatest values of the corresponding parameters in the matrices A and B, and a_i must be replaced by $a_i - \lambda b_i$). In both cases the polynomials $\psi(z)$, $\varphi_i(z)$ and the matrices $\tilde{T}$ and V_s depend on λ.

In the next section, we modify Algorithm 3 in order to compute the ratio $p(\lambda)/p'(\lambda)$, where $p(\lambda) = \det(A - \lambda I)$, with increasing the asymptotic cost only by a constant factor. In this case, we let λ vary (instead of setting $\lambda = 0$), and we replace a_0 by $a_0 - \lambda$ in the definition of $\varphi(z)$, so that $\varphi(z)$ turns into $\varphi(z, \lambda) = \sum_{i=0}^{k} z^i a_{i-r} - \lambda z^r$, and the derivative of φ with respect to λ is $-z^r$.

5. Extension to Computing the Ratio $p(\lambda)/p'(\lambda)$ and to the Case of Block Matrices

Computing the ratio $p(\lambda)/p'(\lambda)$, we will follow the approach of Bini and Pan (1988). We recall the identity

$$(\det X)' = \operatorname{trace}(\operatorname{Adj}(X)X'),$$

where $\operatorname{Adj}(X)$ is the adjugate (adjoint) matrix of a matrix X, and X' is the matrix whose (i, j)-entry is the first derivative of the (i, j)-entry of X. Thus, $\det X/(\det X)' = 1/\operatorname{trace}(X^{-1}X')$. From (3), we have

$$p(\lambda)/p'(\lambda) = \det V_s/(\det V_s)'$$

Therefore, the problem is reduced to the computation of the Toeplitz matrices V_s and V_s', or, equivalently, of the vectors $\mathbf{u}$ and $\mathbf{u}'$ (note that all the derivatives are with respect to the variable λ and not z.)

Now, taking derivatives at stage 2 of Algorithm 3, we obtain that

$$\tilde{T}\mathbf{u}' = a_s^{-1}\boldsymbol{\theta}' - \tilde{T}'\mathbf{u},$$

so that the evaluation of $\mathbf{u}'$, and therefore, of V_s', is reduced to:

(a) computing the vector $\boldsymbol{\theta}'$;

(b) computing the product $\tilde{T}'\mathbf{u}$, which amounts to the shift and scaling of the vector $\mathbf{u}$ (in fact, by definition, only the sth diagonal below the principal diagonal depends on λ, therefore the matrix $\tilde{T}'$ only has nonzero entries on the sth diagonal filled with a_s^{-1});

(c) solving a lower triangular $k \times k$ Toeplitz system.

Adding few more operations at stage 1 of Algorithm 3, we may compute the coefficients θ_i' of the derivative of the polynomial $\psi' = \sum_{i=0}^{k-1} \theta_i' z^i$. Indeed, taking derivatives at stage 1 of Algorithm 3 (recall, from the end of Section 4, that the derivative of φ with respect to λ is $-z^r$), we obtain that

$$\psi_{i+1}' = 2\psi_i\psi_i' - Q_i z^r \bmod \varphi, \qquad i = 0, \ldots, h-1,$$

where Q_i is the quotient and ψ_{i+1} the remainder of the division of ψ_i^2 and φ, i.e., $\psi_i^2 = Q_i\varphi + \psi_{i+1}$. For a fixed i, assuming the coefficients of ψ_i, ψ_i', and Q_i precomputed, we may evaluate the coefficients of ψ_{i+1}' by using one polynomial multiplication and one polynomial division. When we arrive at the coefficients of $\psi' = \psi_h'$, the computational cost of stage 1 of Algorithm 3 is roughly doubled. Thus, the evaluation of the ratio $p(\lambda)/p'(\lambda)$ can be carried out for the cost that roughly doubles the cost of the evaluation of $p(\lambda)$.

In the case where T is an $n \times n$ block matrix with $m \times m$ blocks $t_{j-i} \in \mathbf{C}^{m\times m}$, Algorithm 3 can be extended by means of the technique used in Bini and Pan (1988). The main difficulty of dealing with block entries is that the entries no more belong to a number field but to a noncommutative algebra. Thus, we have to take care of commutativity and of nonsingularity of the blocks. The transition from scalars to blocks leads to matrix-difference equations, which can be solved in terms of block Frobenius matrices by using matrix polynomials, that is, polynomials whose coefficients are matrices. As is shown in Bini and Pan (1988), the first stage of Algorithm 3 still applies to block matrices, provided that we only use the "right" quotients and the right remainders when we perform matrix-polynomial division; that is, in the matrix-polynomial division, the divisor is kept on the right of the quotient. The extensions of stages 2 and 3 are straightforward; it is also easy to verify the extension of the identity $F^i\mathbf{e}_k = \tilde{T}^{-1}\mathbf{e}_{k-i}$ to the block case.

By using the results of Section 2 we deduce that, in the case of block martices, the cost of Algorithm 3 is given by $O(m^3k \log n + m^2k \log k \log n + m^3k^2 \log k)$ arithmetic operations, which can be reduced to $O(m^3k \log n + m^2k \log k \log n + m^3k \log^2 k)$ if the block Toeplitz matrix V_s has a block LU factorization.

6. Concluding Remarks

(1) If k is small, then the asymptotically fast auxiliary algorithms that support the bounds of Tables I and II of Section 2 become inferior to straightforward algorithms, which should be used in our computations in this case with the respective changes of the overall cost bounds, and similarly in the case where the computations are performed over the fields of constants that do not support fast Fourier transform, as well as where we apply the superfast algorithms of Canny *et al.* (1989) (dealing with $k \times k$ transposed Vandermonde matrices) and of Ammar and Gragg (1987) (computing the determinant of $k \times k$ Toeplitz matrices). The latter algorithms only involve $O(k \log^2 k)$ arithmetic operations but for smaller k they are inferior to the algorithm of Golub and van Loan (1989, p. 122) and Del Sarte and Genin (1987), respectively, which use roughly $2.5k^2$ and $2k^2$ arithmetic operations, respectively. (In fact the $O(k \log^2 k)$ algorithm of Canny *et al.*, 1989, is nonpractical due to the problems of numerical stability.)

(2) Of our two sequential algorithms 2 and 3 for computing $p(\lambda) = \det(T - \lambda I)$, Algorithm 2 computes an approximation to $p(\lambda)$ in $O(k \log n + k^2 \log k \log(bk))$ arithmetic operations (where b measures the absolute precision of the approximation to the zeros of an auxiliary kth degree polynomial), and Algorithm 3 exactly computes $p(\lambda)$ (in infinite precision arithmetic) in $O(k \log k \log n)$ arithmetic operations. Algorithm 3 can be extended to the computation of the ratio $p(\lambda)/p'(\lambda)$ and to the case of block matrices. Due to their different natures, a more fair comparison of the costs of these two algorithms could be made under the Boolean model of computation in terms of their bit-cost, where we should consider the number of Boolean operations involved as a function in n, k and the prescribed number of correct digits in the output. Such an analysis would be closely related to the study of the numerical stability of the algorithms.

(3) Another interesting open problem is to give a nontrivial lower bound on the complexity of the computation of $\det A$ for a banded Toeplitz matrix A under the sequential arithmetic model of computation. Since $\det A$ is a multivariate polynomial of degree n in $k + 1$ variables, a trivial lower bound on the complexity is $\log_2 n$ (compare Borodin and Monro, 1975). This lower bound holds under the sequential arithmetic model where only arithmetic operations are allowed as the unit cost operations. We observe that the arithmetic complexity of the solution can even be made constant if we also assign the unit cost to more complicated computations such as exponentiation of the variables to the nth powers or computation of logarithmic and trigonometric functions of complex variable depending on n (see Trench, 1991a). This fact again suggests that the

Boolean model should be more appropriate in order to analyze the complexity of the problem.

REFERENCES

AMMAR, G. S., AND GRAGG, W. G. (1987), "The generalized Schur Algorithm for the Superfast Solution of Toeplitz Systems," Lecture Notes in Mathematics, Vol. 1237, pp. 315–330, Springer-Verlag, Berlin.

AHO, A. V., HOPCROFT, J. E., AND ULLMAN, J. D. (1976), "The Design and Analysis of Computer Algorithms," Addison-Wesley, Reading, MA.

BINI, D., AND CAPOVANI, M. (1983a), Fast parallel and sequential computations and spectral properties concerning band Toeplitz matrices, *Calcolo* **20,** 177–189.

BINI, D., AND CAPOVANI, M. (1983b), Spectral and computational properties of band symmetric Toeplitz matrices, *Linear Algebra Appl.* **52,** 99–126.

BINI, D., AND CAPOVANI, M. (1987), Tensor rank and border rank of band Toeplitz matrices, *SIAM J. Comput.* **16,** 252–258.

BINI, D. (1984), Parallel solution of certain Toeplitz linear systems, *SIAM J. Comput.* **13,** 268–276.

BINI, D. (1985), Tensor and border rank of certain classes of matrices and the fast evaluation of determinant, inverse matrix and eigenvalues, *Calcolo* **22,** 209–227.

BINI, D., AND PAN, V. (1988), Efficient algorithms for the evaluation of the eigenvalues of (block) banded Toeplitz matrices, *Math. Comp.* **50,** 431–448.

BINI, D., AND PAN, V. (1986), Polynomial division and its computational complexity, *J. Complexity* **2,** 179–203.

BINI, D., AND PAN, V. (1990), Parallel complexity of tridiagonal symmetric eigenvalue problem, *in* "Proc. 2nd Ann. ACM–SIAM Symp. on Discrete Algorithms," pp. 384–393.

BINI, D., AND PAN, V. (1991), "Numerical and Algebraic Computations with Matrices and Polynomials" (R. V. Book, Ed.), Lecture Notes in Theoretical Computer Science, Vols. 1 and 2, Birkhäuser, Boston.

BORODIN, A., VON ZUR GATHEN, J., AND HOPCROFT, J. (1982), The parallel matrix and GCD computations, *Inform. and Control* **52**(3), 241–256.

BORODIN, A., AND MUNRO, I. (1975), "The Computational Complexity of Algebraic and Numeric Problems," Elsevier, New York.

CANNY, J., KALTOFEN, E., AND LAKSHMAN, Y. (1989), Solving systems of nonlinear polynomial equations faster, *in* "Proc. ACM-SIGSAM ISSAC-89," pp. 121–128.

DEL SARTE, P., AND GENIN, Y. (1987), On the splitting of classical algorithms in linear prediction theory, *IEEE Trans. Acoust. Speech Signal Process. ASSP–35* **5,** 645–653.

EPPSTEIN, D., AND GALIL, Z. (1988), Parallel algorithmic techniques for combinatorial computation, *Ann. Rev. Comput. Sci.* **3,** 233–283.

GOLUB, G. H., AND VAN LOAN, C. F. (1989), "Matrix Computations," Johns Hopkins Univ. Press, Baltimore, MD.

KARP, R. M., AND RAMACHANDRAN, V. (1990), Parallel algorithms for shared-memory machines, *in* "Handbook for Theoretical Computer Science," pp. 869–941, North-Holland, Amsterdam.

PAN, V. (1987), Sequential and parallel complexity of approximate evaluation of polynomial zeros, *Comput. Math. Appl.* **14,** 8, 591–622.

PAN, V. (1991), Parametrization of Newton's iteration for computation with structured matrices and applications, *Comput. Math. Appl.*, to appear.

PAN, V. (1984), How to multiply matrices faster, *in* "Springer's Lecture Notes in Computer Science," Vol. 179, 212 pp.

REIF, J., AND TATE, S. (1989), Optimal size division circuits, *in* "Proc. 21st Ann. ACM Symp. Theory of Computing," pp. 264–270.

RENEGAR, J. (1987), On the worst-case complexity of approximating zeros of polynomials, *J. Complexity* **3**(2), 90–113.

SMALE, S. (1986), Newton's method estimates from data at one point, *in* "The Meeting of Disciplines. New Directions in Pure, Applied and Computational Mathematics" (R. Ewing, K. Gross, and C. Martin, Eds.), Springer, New York.

STOER, J., AND BULIRSCH, R. (1980), "Introduction to Numerical Analysis," Springer, Berlin.

TISMENETSKY, M. (1987), Determinant of block-Toeplitz band matrices, *Linear Algebra Appl.* **85,** 165–184.

TRENCH, W. F. (1991a), Characteristic polynomials of symmetric rationally generated Toeplitz matrices, *Linear and Multilinear Algebra*, to appear.

TRENCH, W. F. (1991b), Numerical solution of the eigenvalue problem for rationally generated Toeplitz matrices, *Linear Algebra Signals Systems and Control*, to appear.

TRENCH, W. F. (1985), On the eigenvalue problem for Toeplitz band matrices, *Linear Algebra Appl.* **64,** 199–214.